IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

| | | | |
|---|---|---|---|
| Applicant | : Gilbert Wolrich et al. | Art Unit | : 2183 |
| Assignee | : Intel Corporation | Examiner | : Daniel H. Pan |
| Serial No. | : 10/070,091 | Conf. No: | : 7309 |
| Filed | : February 27, 2002 | | |

Title : REGISTER SET USED IN MULTITHREADED PARALLEL PROCESSOR ARCHITECTURE

**MAIL STOP APPEAL BRIEF – PATENTS**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF ON BEHALF OF
GILBERT WOLRICH, MATTHEW J. ADILETTA, WILLIAM R. WHEELER, DEBRA BERNSTEIN, AND DONALD F. HOOPER

The fees in the amount of $500 are being paid concurrently on the Electronic Filing System (EFS) by way of Deposit Account authorization. Please apply any other required fees to deposit account 06-1050, referencing the attorney docket number shown above.

**(i.) Real Party In Interest**

The real party in interest in the above application is Intel Corporation.

**(ii.) Related Appeals and Interferences**

The Appellant is not aware of any appeals or interferences related to the above-identified patent application.

**(iii.) Status of Claims**

This is an appeal from the decision of the Primary Examiner in a Final Office Action dated February 20, 2007, rejecting claims 1-8, and 10-21, all of the claims of the above application. The claims have been twice rejected. Claims 1-8 and 10-21 are the subject of this appeal.

**(iv.) Status of Amendments**

All amendments have been entered. Appellant filed a Notice of Appeal on May 21, 2007.

**(v.) Summary of Claimed Subject Matter**

Claim 1

One aspect of Appellant's invention is set out in claim 1, as a method of maintaining execution threads in a parallel multithreaded processor. ***"Referring to FIG. 1, a communication system 10 includes a parallel, hardware-based multithreaded processor 12."***[1] ***"By employing hardware context swapping within each of the microengines 22a-22f, the hardware context swapping enables other contexts with unique program counters to execute in that same microengine. Thus, another thread e.g., Thread_1 can function while the first thread, e.g., Thread_0, is awaiting the read data to return."*** [2]

Inventive features of Appellant's claim 1 include accessing, by a thread executing in the multithreaded processor, a register in a register set organized into a plurality of windows of registers. ***"As will be described in FIG. 6 [sic – should be FIG. 5], in this implementation***

[1] [Specification, page 1, lines 28-29].
[2] [Specification, page 3, lines 12-16]

***there are 64 general purpose registers in a first bank, Bank A and 64 in a second bank, Bank B. The general purpose registers are windowed as will be described so that they are relatively and absolutely addressable."*** [3] Appellant's FIG. 5 shows exemplary partitioning of registers banks into a plurality of windows of registers.

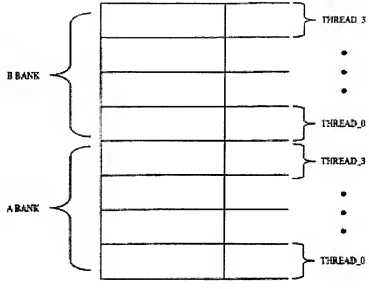



FIG. 5

A further inventive feature of Appellant's claim 1 includes each of the plurality of windows of registers associated with a corresponding thread, each register in the plurality of windows of registers being relatively addressable by the corresponding thread. ***"Across banks A and B, the register set 76b is also organized into four windows 76b$_0$-76b$_3$ of 32 registers that are relatively addressable per thread. Thus, thread_0 will find its register 0 at 77a (register 0), the thread_1 will find its register_0 at 77b (register 32), thread_2 will find its register_0 at 77c (register 64), and thread_3 at 77d (register 96)."*** [4]

[3] [Id. page 9, lines 13-16].
[4] [FIG. 5, and Specification, page 10, lines 15-19]

Another inventive feature of Appellant's independent claim 1 includes each register being absolutely addressable by two or more of the threads executing on the multithreaded processor with absolutely addressable comprises providing an exact address of the register with the exact address specified in an instruction. ***"Absolute addressing is also supported in this architecture where any one of the absolute registers may be accessed by any of the threads by providing the exact address of the register. Addressing of general purpose registers 78 occurs in 2 modes depending on the microword format. The two modes are absolute and relative. In absolute mode, addressing of a register address is directly specified in 7-bit source field (a6-a0 or b6-b0):"***[5]

Claim 12

Claim 12 recites another aspect of the invention. Claim 12 is directed to a hardware based multi-threaded processor that comprises a processor unit comprising control logic circuit including context event switching logic, the context switching logic arbitrating access to the microengine for a plurality of executable threads, and an arithmetic logic unit to process data for executing threads. ***"Referring to FIG. 1, a communication system 10 includes a parallel, hardware-based multithreaded processor 12."***[6] Further, FIG. 3 is a block diagram of an exemplary embodiment of a microengine functional unit employed in the hardware-based multithreaded processor.

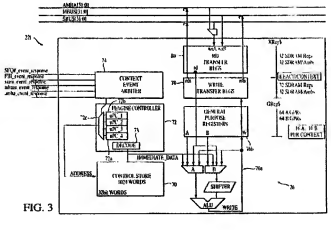


FIG. 3

[5] [Id., page 10, lines 28-33]

***"Referring to FIG. 3, an exemplary one of the microengines 22a-22f, e.g., microengine 22f is shown. The microengine includes a control store 70 which, in one implementation, includes a RAM of here 1,024 words of 32 bit. The RAM stores a microprogram. The microprogram is loadable by the core processor 20. The microengine 22f also includes controller logic 72. The controller logic includes an instruction decoder 73 and program counter (PC) units 72a-72d. The four micro program counters 72a-72d are maintained in hardware. The microengine 22f also includes context event switching logic 74."***[7] ***"The context event logic 74 has arbitration for the four (4) threads. In one embodiment, the arbitration is a round robin mechanism. Other techniques could be used including priority queuing or weighted fair queuing. The microengine 22f also includes an execution box (EBOX) data path 76 that includes an arithmetic logic unit 76a and general purpose register set 76b. The arithmetic logic unit 76a performs arithmetic and logical functions as well as shift functions."***[8]

Another inventive feature of Appellant's claim 12 includes a register set that is organized into a plurality of windows of registers. ***"As will be described in FIG. 6 [sic – should be FIG. 5], in this implementation there are 64 general purpose registers in a first bank, Bank A and 64 in a second bank, Bank B. The general purpose registers are windowed as will be described so that they are relatively and absolutely addressable."***[9]

A further inventive feature of Appellant's claim 12 includes each of the plurality of windows of registers associated with a corresponding one of the plurality of threads, each register in the plurality of windows of registers being relatively addressable by the corresponding thread associated with the respective window of registers. ***"Across banks A and B, the register set 76b is also organized into four windows 76b$_0$-76b$_3$ of 32 registers that are relatively addressable per thread. Thus, thread_0 will find its register 0 at 77a (register 0), the thread_1 will find its register_0 at 77b (register 32), thread_2 will find its register_0 at 77c (register 64), and thread_3 at 77d (register 96)."***[10]

---

[7] [Id., page 8, lines 16-23]
[8] [Id., page 9, lines 7-12]
[9] [Id., page 9, lines 13-16].
[10] [FIG. 5, and Specification, page 10, lines 15-19]

Another inventive feature of Appellant's claim 12 includes each register absolutely addressable by two or more of the threads executing on the multi-threaded processor, with any one of the registers of the register set being absolutely addressable by providing an exact address of the register with the exact address specified in an instruction. ***"Absolute addressing is also supported in this architecture where any one of the absolute registers may be accessed by any of the threads by providing the exact address of the register. Addressing of general purpose registers 78 occurs in 2 modes depending on the microword format. The two modes are absolute and relative. In absolute mode, addressing of a register address is directly specified in 7-bit source field (a6-a0 or b6-b0):"***[11]

Claim 21

Claim 21 recites a further aspect of the invention. Claim 21 is a computer program product residing on a computer readable storage medium for managing execution of multiple threads in a multithreaded processor comprising instructions causing a processor to perform the operations pertaining to the inventive features described below. ***"The hardware-based multithreaded processor 12 also includes a central controller 20 that assists in loading microcode control for other resources of the hardware-based multithreaded processor 12 and performs other general purpose computer type functions such as handling protocols, exceptions, extra support for packet processing where the microengines pass the packets off for more detailed processing such as in boundary conditions. In one embodiment, the processor 20 is a Strong Arm® (Arm is a trademark of ARM Limited, United Kingdom) based architecture. The general purpose microprocessor 20 has an operating system. Through the operating system the processor 20 can call functions to operate on microengines 22a-22f. The processor 20 can use any supported operating system preferably a real time operating system. For the core processor implemented as a Strong Arm architecture, operating systems such as, MicrosoftNT® real-time, VXWorks and □CUS, a freeware operating system available over the Internet, can be used."***[12]

---

[11] [Specification, page 10, lines 28-33]
[12] [Id., page 2, lines 3-15]

An inventive feature of Appellant's independent claim 21 includes instructions to access, by an executing thread in the multithreaded processor, a register in a register set organized into a plurality of windows of registers. ***"Across banks A and B, the register set 76b is also organized into four windows 76b$_0$-76b$_3$ of 32 registers that are relatively addressable per thread."*** [13] Thus, threads access the windows of registers. Further, ***"[a]s will be described in FIG. 6 [sic – should be FIG. 5], in this implementation there are 64 general purpose registers in a first bank, Bank A and 64 in a second bank, Bank B. The general purpose registers are windowed as will be described so that they are relatively and absolutely addressable."*** [14]

A further inventive feature of Appellant's independent claim 21 includes each of the plurality of windows of registers being associated with a corresponding thread, each register in the plurality of windows of registers being relatively addressable by the corresponding thread. ***"Across banks A and B, the register set 76b is also organized into four windows 76b$_0$-76b$_3$ of 32 registers that are relatively addressable per thread. Thus, thread_0 will find its register 0 at 77a (register 0), the thread_1 will find its register_0 at 77b (register 32), thread_2 will find its register_0 at 77c (register 64), and thread_3 at 77d (register 96)."*** [15]

Another inventive feature of Appellant's independent claim 21 includes each register absolutely addressable by two or more threads executing on the multithreaded processor with absolutely addressable comprises instructions that when executed cause the processor to provide an exact address of the register with the exact address specified in an instruction. ***"Absolute addressing is also supported in this architecture where any one of the absolute registers may be accessed by any of the threads by providing the exact address of the register. Addressing of general purpose registers 78 occurs in 2 modes depending on the microword format. The two modes are absolute and relative. In absolute mode, addressing of a register address is directly specified in 7-bit source field (a6-a0 or b6-b0):"*** [16]

---

[13] [Id., page 10, lines 15-16]
[14] [Id., page 9, lines 13-16]
[15] [FIG. 5, and Specification, page 10, lines 15-19]
[16] [Specification, page 10, lines 28-33]

**(vi.) Grounds of Rejection to be Reviewed on Appeal**

Claims 1 and 21 stand rejected under 35 U.S.C. §101.

Claims 1-6, 8, 10-18, 20 and 21 stand rejected under 35 U.S.C. §102(b) as being anticipated by U.S. Patent No. 5,900,025 to Sollars.

Claims 7 and 19 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Sollars in view of U.S. Patent No. 5,870,597 to Panwar *et al.*

Claims 1-8 and 10-21 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Panwar *et al.* in view of U.S. Patent No. 5,996,068 to Dwyer.

**(vii.) Argument**

Patentable Subject-Matter

The Federal Circuit, in *State Street Bank & Trust Co. v. Signature Financial Group Inc.*, 149 F. 3d 1368, 1374, 47 USPQ2d 1596, 1601-02 (Fed. Cir. 1998), stated:

> After Diehr and Chakrabarty, the Freeman-Walter-Abele test has little, if any, applicability to determining the presence of statutory subject matter. As we pointed out in Alappat, 33 F.3d at 1543, 31 USPQ2d at 1557, application of the test could be misleading, because a process, machine, manufacture, or composition of matter employing a law of nature, natural phenomenon, or abstract idea is patentable subject matter even though a law of nature, natural phenomenon, or abstract idea would not, by itself, be entitled to such protection.[6] The test determines the presence of, for example, an algorithm. Under Benson, this may have been a sufficient indicium of nonstatutory subject matter. However, after Diehr and Alappat, the mere fact that a claimed invention involves inputting numbers, calculating numbers, outputting numbers, and storing numbers, in and of itself, would not render it nonstatutory subject matter, unless, of course, its operation does not produce a "useful, concrete and tangible result." Alappat, 33 F.3d at 1544, 31 USPQ2d at 1557.[7] (Footnotes omitted.)

As to what constitutes a "useful, concrete and tangible result," MPEP 2106.IV.C.2(2) provides:

> **(2) Practical Application That Produces a Useful, Concrete, and Tangible Result**
>
> ...
>
> If USPTO personnel determine that the claim does not entail the transformation of an article, then USPTO personnel shall review the claim to determine it produces a useful, tangible, and concrete result. In making this determination, the focus is not on whether the steps taken to achieve a particular result are useful, tangible, and concrete, but rather on whether the final result achieved by the claimed invention is "useful, tangible, and concrete." In other words, the claim must be examined to see if it includes anything more than a 35 U.S.C. **101** judicial exception. If the claim is directed to a practical application of a 35 U.S.C. **101** judicial exception, USPTO personnel must then determine whether the claim preempts the judicial exception. If USPTO personnel do not find such a practical application, then USPTO personnel have determined that the claim is nonstatutory.
>
> In determining whether a claim provides a practical application of a 35 U.S.C. **101** judicial exception that produces a useful, tangible, and concrete result, USPTO personnel should consider and weigh the following factors:
>
> **a) "USEFUL RESULT"**
>
> For an invention to be "useful" it must satisfy the utility requirement of section **101**. The USPTO's official interpretation of the utility requirement provides that the utility of an invention has to be (i) specific, (ii) substantial and (iii) credible.
>
> ...
>
> **b) "TANGIBLE RESULT"**
>
> The tangible requirement does not necessarily mean that a claim must either be tied to a particular machine or apparatus or must operate to change articles or materials to a different state or thing. However, the tangible requirement does require that the claim must recite more than a 35 U.S.C. **101** judicial exception, in that the process claim must set forth a practical application of that judicial exception to produce a real-world result.
>
> ...
>
> In other words, the opposite meaning of "tangible" is "abstract."
>
> **c) "CONCRETE RESULT"**
>
> Another consideration is whether the invention produces a "concrete" result. Usually, this question arises when a result cannot be assured. In other words, the process must have a result that can be substantially repeatable or the process must substantially produce the same result again.
>
> ... (emphasis added)

With respect to computer program products, MPEP 2106.01(I) explains:

> **I. FUNCTIONAL DESCRIPTIVE MATERIAL: "DATA STRUCTURES" REPRESENTING DESCRIPTIVE MATERIAL PER SE OR COMPUTER PROGRAMS REPRESENTING COMPUTER LISTINGS PER SE**
>
> Data structures not claimed as embodied in computer-readable media are descriptive material *per se* and are not statutory because they are not capable of causing functional change in the computer. See, e.g., *Warmerdam*, 33 F.3d at 1361, 31 USPQ2d at 1760 (claim to a data structure *per se* held nonstatutory). Such claimed data structures do not define any structural and functional interrelationships between the data structure and other claimed aspects of the invention which permit the data structure's functionality to be realized. In contrast, a claimed computer-readable medium encoded with a data structure defines structural and functional interrelationships between the data structure and the computer software and hardware components which permit the data structure's functionality to be realized, and is thus statutory.
>
> Similarly, computer programs claimed as computer listings *per se*, i.e., the descriptions or expressions of the programs, are not physical "things." They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer which permit the computer program's functionality to be realized. In contrast, a claimed computer-readable medium encoded with a computer program is a computer element which defines structural and functional interrelationships between the computer program and the rest of the computer which permit the computer program's functionality to be realized, and is thus statutory. See *Lowry*, 32 F.3d at 1583-84, 32 USPQ2d at 1035. Accordingly, it is important to distinguish claims that define descriptive material *per se* from claims that define statutory inventions.

Anticipation

"It is well settled that anticipation under 35 U.S.C. §102 requires the presence in a single reference of all of the elements of a claimed invention." *Ex parte Chopra*, 229 U.S.P.Q. 230, 231 (BPA&I 1985) and cases cited.

"Anticipation requires the presence in a single prior art disclosure of all elements of a claimed invention arranged as in the claim." *Connell v. Sears, Roebuck & Co.*, 220 U.S.P.Q. 193, 198 (Fed. Cir. 1983).

"This court has repeatedly stated that the defense of lack of novelty (i.e., 'anticipation') can only be established by a single prior art reference which discloses each and every element of

the claimed invention." *Structural Rubber Prod. Co. v. Park Rubber Co.*, 223 U.S.P.Q. 1264, 1270 (Fed. Cir. 1984), citing five prior Federal Circuit decisions since 1983 including *Connell.*

In a later analogous case the Court of Appeals for the Federal Circuit again applied this rule in reversing a denial of a motion for judgment n.o.v. after a jury finding that claims were anticipated. *Jamesbury Corp. v. Litton Industrial Prod., Inc.*, 225 U.S.P.Q. 253 (Fed. Cir. 1985). After quoting from *Connell*, "Anticipation requires the presence in a single prior art disclosure of all elements of a claimed invention arranged as in the claim," 225 U.S.P.Q. at 256, the court observed that the patentee accomplished a constant tight contact in a ball valve by a lip on the seal or ring which interferes with the placement of the ball. The lip protruded into the area where the ball will be placed and was thus deflected after the ball was assembled into the valve. Because of this constant pressure, the patented valve was described as providing a particularly good seal when regulating a low pressure stream. The court quoted with approval from a 1967 Court of Claims decision adopting the opinion of then Commissioner and later Judge Donald E. Lane:

> [T]he term "engaging the ball" recited in claims 7 and 8 means that the lip contacts the ball with sufficient force to provide a fluid tight seal **** The Saunders flange or lip only sealingly engages the ball 1 on the upstream side when the fluid pressure forces the lip against the ball and never sealingly engages the ball on the downstream side because there is no fluid pressure there to force the lip against the ball. The Saunders sealing ring provides a compression type of seal which depends upon the ball pressing into the material of the ring. *** The seal of Saunders depends primarily on the contact between the ball and the body of the sealing ring, and the flange or lip sealingly contacts the ball on the upstream side when the fluid pressure increases. 225 U.S.P.Q. at 258.

Relying on *Jamesbury*, the ITC said, "Anticipation requires looking at a reference, and comparing the disclosure of the reference with the claims of the patent in suit. A claimed device is anticipated if a single prior art reference discloses all the elements of the claimed invention as arranged in the claim." *In re Certain Floppy Disk Drives and Components Thereof*, 227 U.S.P.Q. 982, 985 (U.S. ITC 1985).

Obviousness

"It is well established that the burden is on the PTO to establish a prima facie showing of obviousness, *In re Fritsch*, 972 F.2d. 1260, 23 U.S.P.Q.2d 1780 (C.C.P.A., 1972)."

In *KSR International Co. v. Teleflex Inc.*, __ U.S. __, 2007 WL 1237837 (Apr. 30, 2007), the Supreme Court reversed the Court of Appeal's for the Federal Circuit's decision that reversed a summary judgment of obviousness on the ground that the district court had not adequately identified a motivation to combine two prior art references. The invention was a combination of a prior art repositionable gas pedal, with prior art electronic (rather than mechanical cable) gas pedal position sensing. The Court first rejected the "rigid" teaching suggestion motivation (TSM) requirement applied by the Federal Circuit, since the Court's obviousness decisions had all advocated a "flexible" and "functional" approach that cautioned against "granting a patent based on the combination of elements found in the prior art."

With respect to the genesis of the TSM requirement, the Court noted that although "[a]s is clear from cases such as Adams[17], a patent composed of several elements is not proved obvious merely by demonstrating that each of its elements was, independently, known in the prior art. Although common sense directs one to look with care at a patent application that claims as innovation the combination of two known devices according to their established functions, it can be important to identify a reason that would have prompted a person of ordinary skill in the relevant field to combine the elements in the way the claimed new invention does. This is so because inventions in most, if not all, instances rely upon building blocks long since uncovered, and claimed discoveries almost of necessity will be combinations of what, in some sense, is already known."

In application of the TSM requirement, the Court cautioned that: "Helpful insights, however, need not become rigid and mandatory formulas; and when it is so applied, the TSM test is incompatible with our precedents." To the extent the Fed Cir has been applying a flexible rule recently, that flexible rule was not applied in this case, and the Fed Cir can figure out how to match its actions to this decision.

[17] United States v. Adams, 383 U. S. 39, 40 (1966)

"The mere fact that the prior art could be so modified would not have made the modification obvious unless the prior art suggested the desirability of the modification." *In re Gordon*, 221 U.S.P.Q. 1125, 1127 (Fed. Cir. 1984).

> Although the Commissioner suggests that [the structure in the primary prior art reference] could readily be modified to form the [claimed] structure, "[t]he mere fact that the prior art could be so modified would not have made the modification obvious unless the prior art suggested the desirability of the modification." *In re Laskowski*, 10 U.S.P.Q. 2d 1397, 1398 (Fed. Cir. 1989).

"The claimed invention must be considered as a whole, and the question is whether there is something in the prior art as a whole to suggest the desirability, and thus the obviousness, of making the combination." *Lindemann Maschinenfabrik GMBH v. American Hoist & Derrick*, 221 U.S.P.Q. 481, 488 (Fed. Cir. 1984).

> Obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination. Under Section 103, teachings of references can be combined only if there is some suggestion or incentive to do so. *ACS Hospital Systems, Inc. v. Montefiore Hospital*, 221 U.S.P.Q. 929, 933 (Fed. Cir. 1984) (emphasis in original, footnotes omitted).

"The critical inquiry is whether 'there is something in the prior art as a whole to suggest the desirability, and thus the obviousness, of making the combination.'" *Fromson v. Advance Offset Plate, Inc.*, 225 U.S.P.Q. 26, 31 (Fed. Cir. 1985).

**(1) Claims 1 and 21 Are Directed to Patentable Subject Matter**

The Examiner contends that claims 1 and 21 are unpatentable under 35 U.S.C. §101 on the ground that the claimed invention is allegedly directed to non-statutory subject matter.

Specifically, the Examiner stated:

> 10. As to the newly amended claim 21, 21 is not limited to tangible embodiments. In view of applicant's disclosure, specification page ( 2), line ( 3-15), the medium is not limited to tangible embodiments, instead being defined as including both tangible embodiments (e.g., ( hardware based processor)) and intangible embodiments (e.g., (internet). Although the claim recite the instructions cause the access of absolutely address and relatively address, based on broadest interpretation, it is read as intended use, not a positive limitation. The focus is not on the steps or feature taken to achieve the final result which is useful, tangible, and concrete, but rather the final result achieved which is useful, tangible, and concrete (see MPEP 2100, 101 Interim Guidelines published at uspto.gov). No final result which is useful, tangible, and concrete can be found in the claims. Therefore, it is directed non-statutory subject matter. Although applicant recites the a computer program product residing one computer readable storage medium, no components of the program product, nor the detailed elements of the multithreaded processor, which impart the functionalities of the multithreaded processor can be found in the newly amended claim. No practical application of accessing the register in the register windows is not clear. Therefore, no substantial practical application can be found. (Final Action, page 4)

Appellant contends that the Examiner's rejection of claims 1 and 21 as allegedly being directed to non-statutory subject matter is improper.

Appellant's independent claim 1 is directed a method of maintaining execution threads in a parallel multithreaded processor and includes the physical and tangible step of accessing, by a thread executing in the multithreaded processor, a register in a register set.

Claim 21 is directed to "A computer program product residing on a computer readable storage medium for managing execution of multiple threads in a multithreaded processor." Claim 21 recites instructions that cause a processor to access, by an executing thread in the multithreaded processor, a register in a register set.

Appellant's claims 1 and 21 are each useful in that accessing a register in a register set organized into a plurality of windows of register creates a specific, substantive and credible result.

Appellant's claims are tangible in that the claims are all tied to a machine, namely, a multithreaded processor. Further, the multithreaded processor includes a plurality of windows of registers, all of which are tangible, in which registers in those windows are relatively addressable by a corresponding thread, and are also absolutely addressable by two or more threads. Moreover, the tangible multithreaded processor recited in Appellant's claims is one in which the

absolutely addressable feature is performed by providing the register's exact address in an instruction.

Appellant's claims are also concrete because the recited accessing operation (or instructions to access) is repeatable and produces the same results upon different performances of the recited operation (e.g., accessing a register arranged in a window of register by using relative addressing or absolute addressing). As such, applicant's claimed subject matter provides a concrete result as opposed to an abstract result.

Thus, Appellant's independent claims 1 and 21 are directed to an accessing operation that is useful, tangible and concrete.

Appellant's further notes the recited access operations are also final results in that the operations recited in the independent claims achieve a "final" result of accessing a desired register. Once the desired register has been accessed, additional operations that achieve additional results may be performed.

In view of the foregoing, Appellant contends that, contrary to the Examiner's assertions, independent claims 1 and 21 are directed to statutory subject-matter.

**(2) Claims 1-6, 8, 11-18, 20 and 21 are patentable Over the Prior Art**

For the purposes of this appeal only, claims 1-6, 8, 11-18, 20 and 21 stand or fall together.

Claim 1 is representative of this group of claims. Claim 1 is directed to a method for maintaining execution threads in a parallel multithreaded processor that includes the features of "each register in the plurality of windows of registers being relatively addressable by the corresponding thread and absolutely addressable by two or more of the threads executing on the multithreaded processor."

Claim 1 is not Anticipated by Sollars

The Examiner rejected claim 1 as allegedly being anticipated by Sollars. Specifically, the Examiner stated:

35. As to the newly amended claims 1,5,12,20, 21, Sollars disclosed maintaining execution thread in a parallel multithreaded processor (see the concurrent execution of eight threads in col.2, lines 1-14) comprising: a) accessing, by an executing thread in the multithreaded processor, a register set organized into a plurality of relatively addressable windows of registers that are relatively addressable per thread (see the virtually/physically addressable operand register sets in col.5, lines 20-31, see also the partitioned control register subsets in col.5, lines 32-54, col.6, lines 36-67, col.7, lines 1-7, col.7, lines 38-67, col.8, lines 1-11, see also fig.3 and fig.4 for corresponding virtual/physical addressable registers sets, see how register file [register set] organized into register sets [102][104][106] [register windows], see also how the register set 102 further organized into subsets 108) wherein accessing absolutely any one of the relatively (see virtually addressable) and absolutely addressable (see physically addressable) registers comprised providing an exact address of the register (see register addresses in col.8, line5-11), the exact address specified in an instruction associated with the thread (see the operand register sets accessible by the instruction in col.5, lines 25-26, see also the control registers accessible by instruction in col.5, lines 48-51).
36. As to the feature of register absolutely addressable by one or more threads, Sollars taught accessing absolutely any one of the relatively (see virtually addressable) and absolutely addressable (see physically addressable) registers comprised providing an exact address of the register by one or more threads (see register addresses in col.8, lines 5-11, see the concurrent execution of eight threads in col.2, lines 1-14). [Final Action, pages 9-10]

Further, the Examiner stated in the April 26, 2007, Advisory Action, "[a]s for Sollars, Sollars taught allocation of register set 106 to a new thread (see col. 12, lines 59-67, col. 13, lines 1-11). Therefore the register set 106 was addressable by the two or more threads as claimed." [Advisory Action, page 2]

Appellant contends that the Sollars reference fails to disclose all the features of independent claim 1, particularly the features of "each register in the plurality of windows of registers being relatively addressable by the corresponding thread and absolutely addressable by two or more of the threads executing on the multithreaded processor."

Sollars describes a processor with a number of control registers logically organized in a hierarchical manner to control the processor, context and threads executed by Sollars' processor (see, for example, the Abstract and col. 1, line 60 to col. 2, line 6). Specifically, Sollars explains that "[p]rimary control register file 20a comprises a plurality of control registers for performing the conventional functions of storing control and status information of executing processes (col. 5, lines 32-35) and describes the register file as follows:

**FIG. 2 illustrates a logical view of the control registers under the presently preferred embodiment of the present invention. As shown, control registers of primary control register file 20a are organized into registers sets 102, 104, and 106, which in turn are organized into a hierarchy having three control register levels, i.e. a system level, a context level, and a thread level. At the highest level is a set of control registers 102 for controlling overall system operation. At the second highest level are multiple sets of control registers 104 for controlling concurrent execution of processes in multiple peer contexts. At the third level are multiple sets of control registers 106 for controlling concurrent execution of multiple peer process threads of the concurrently executing contexts. (FIG. 2 and col. 6, lines 35-49)**

Sollars' FIG. 2 is provided below:

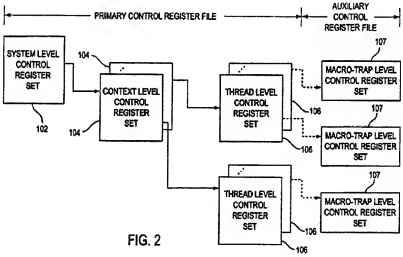



FIG. 2

Sollars further explains:

> **In one implementation of the preferred embodiment, eight sets of context control registers are provided for concurrently supporting up to eight active contexts, and 64 sets of thread control registers are provided for concurrently supporting up to eight active threads for each of the active contexts. Each of the system and context control register sets comprises 32 control registers, whereas each thread control register set comprises 16 control registers. (col. 2, lines 7-14)**

Thus, control register sets each support a corresponding context.

Similarly, different threads associated with a particular context access only their own respective allocated thread control register sets. Specifically, as Sollars explains in relation to register allocation:

> **As shown in FIG. 12a, similarly, upon receipt of a valid request to create a new thread, i.e. from a thread with the proper context privilege, step 228, the context checks to determine if all thread level control register sets 106 have been allocated, step 230. If all thread level control register sets 106 have been allocated, the context further determines if the execution priority of the "new" thread requested is higher than at least one of the allocated threads, step 232. If the relative priority determination is unfavorable, the context simply queues the new thread request in a thread request queue of processor 10, step 234. On the other hand, if the relative priority determination is favorable, the context deallocates and queues the lowest priority allocated thread, step 236.**
>
> **Upon determining that there is at least one free thread level control register set 106 at step 230, or creating one through step 236, the context allocates a free thread level control register set 106 to the "new" thread, step 238. The context then requests the operating system to cause IFU 12 to reprioritize allocation of its internal resources. (col. 12, line 59 to col. 13, line 11)**

A new thread is allocated a free register set (i.e., a register set that is not being currently used). Moreover, if all registers have been allocated and a request to create a new threads does not have sufficient priority, the request is queued (i.e., the request is not immediately served but rather is deferred until a later time). Accordingly, different threads do not share registers, and certainly do not address registers allocated to some other thread.

Therefore, contrary to the Examiner's contentions, different contexts/threads do not access any register set other than the one they have been allocated. Indeed, because control registers that are assigned to a particular context (or thread) are used primarily to maintain control of that context (or thread)[18], there would be no need for another context (or thread) to access any other control register.

The Examiner's contended, in responding to the Appellant's arguments presented in the Amendment in Reply to Action of September 5, 2006, that:

> 7. As to b), Sollars taught access of the different windows (see the different number of the dynamically associated thread 106 in col.7, lines 25-37), and the concurrent execution of eight threads for each of the active contexts (see col.2, lines 1-14). Therefore, a context control register was accessible by more than one thread. (Final Action, page 2)

The passage in Sollars that the Examiner relies on states:

> **At the second highest level of the logical hierarchy is a number of control register sets for controlling concurrent execution of processes in multiple**

[18] See, for example, Sollars, col. 5, line 32, to col. 6, line 3.

**peer contexts (hereinafter simply contexts). At the third highest level of the logical hierarchy is a number of control register sets for controlling concurrent execution of multiple peer process threads (hereinafter simply threads) for each of the concurrently executing peer contexts.**

**In one implementation of the preferred embodiment, eight sets of context control registers are provided for concurrently supporting up to eight active contexts, and 64 sets of thread control registers are provided for concurrently supporting up to eight active threads for each of the active contexts. Each of the system and context control register sets comprises 32 control registers, whereas each thread control register set comprises 16 control registers. (Sollars, col. 1, line 66, to col. 2, line 14)**

Thus, Sollars indicates that context control registers are provided to control context, and that each context may have up to eight threads that are supported by separate groups of registers. But nothing in the above passage, or elsewhere in Sollars, suggests that threads corresponding to a particular context access the registers provided to support that particular context.

Furthermore, Appellant notes that independent claim 1 requires "each of the plurality of windows of registers associated with a corresponding thread, each register in the plurality of windows of registers being relatively addressable by the corresponding thread and absolutely addressable by two or more of the threads executing on the multithreaded processor with absolutely addressable comprises providing an exact address of the register with the exact address specified in an instruction."

The examiner argues that: "...**Sollars taught access of the different windows (see the different number of the dynamically associated thread 106 in col.7, lines 25-37), and the concurrent execution of eight threads for each of the active contexts (see col.2, lines 1-14). Therefore, a context control register was accessible by more than one thread. (Final Action, page 2)**" Appellant contends that this reasoning is irrelevant to the claimed subject matter. Claim 1 includes the feature that "each of the plurality of windows of registers associated with a corresponding thread." In contrast, by the Examiner's own admission, Sollars' context control registers (which the Examiner uses in supporting his contention that Sollars discloses a register being accessed by more than one thread) is associated with multiple threads rather than with a single thread (which is what claim 1 requires)

Thus, for the foregoing reasons, Sollars fails to disclose or suggest at least the feature of "each register in the plurality of windows of registers being relatively addressable by the

corresponding thread and absolutely addressable by two or more of the threads executing on the multithreaded processor," as required by Appellant's independent claim 1.

Claim 1 is Not Rendered Obvious Over Panwar and Dwyer

The Examiner admitted that "Panwar did not specifically show that the register was addressable by two or more threads executing on the multiprocessor as claimed." Indeed, Panwar discloses a processor that speculatively executes instructions that specify logical addresses, and a processor that converts the logical addresses to physical addresses (Abstract). Particularly, a register set is divided into windows having 32 registers. Different processes or programs executing on a processor 102 can allocate their own independent window (col. 7, lines 43-47). Panwar further discloses that programs executing on a processor 102 access registers through a typical naming convention, such as r0, r1, r2, …, r30 (col. 7, lines 51-54), and states:

> **The five-bit register addresses encoded in an instruction word specify the instruction's source registers and the destination register. These register specifiers are logical addresses that index registers within the current register window. (Col. 2, line 66 – col. 3, line 3)**

Programs executing on Panwar's apparatus therefore use relative (logical) addresses when specifying the desired registers that the programs are to access. At no point does Panwar describe any instruction that specifies a register's absolute address, and Panwar, therefore, does not disclose or suggest at least the features of "each register in the plurality of windows of registers being relatively addressable by the corresponding thread and absolutely addressable by two or more of the threads executing on the multithreaded processor."

The Examiner, however, relied on Dwyer as allegedly disclosing this feature, and stated in the February 20, 2007, Final Action :

> Dwyer taught a system for accessing a register set by a plurality of threads (see the sharing of the register pool for renaming the architectural registers of several threads in col.2 , lines 46-57 ). It would have been obvious to one of ordinary skill in the art to use Dwyer in Panwar for including the register addressable by two or more threads executing on the multiprocessor as clamed because the use of Dwyer could provide Panwar the ability to accept data from more than one thread process, and thereby increasing the ability to adapt to multiple

> bandwidth of maltreated tasks, and it could be achieved by predefining the multiple threads of Dwyer into the configuration file of Panwar with modified control parameters (e.g. the register map with the corresponding threads) so that multiple threads of Dwyer could be recognized by the register of Panwar, and because Panwar also taught a parallel multithreaded processor (see consistent implementation of multithreaded processor 102 in col.6, lines 40-44),which was a suggestion of the need for providing an access to a register, or the like, by a multiplicity of threads, and in doing so, provided a motivation. (Office Action, page 5, paragraph 13)

Additionally, the Examiner stated in the April 26, 2007, Advisory Action that, "[a]s to Dwyer, Dwyer is relatively addressable for the mapping of the registers, and it is absolutely addressable for accessing register set by a plurality of threads (see the sharing of the register pool for renaming the architectural registers of several threads in col. 2, lines 46-57)" [Advisory Action, page 2]

Appellant submits that contrary to the Examiner's contention, Dwyer fails to disclose relatively and absolutely addressable registers, and thus the combination of references relied upon by the Examiner to reject claim 1 fails to disclose at least the features of "each register in the plurality of windows of registers being relatively addressable by the corresponding thread and absolutely addressable by two or more of the threads executing on the multithreaded processor".

Particularly, Dwyer describes an apparatus and scheme for register renaming (col. 2, lines 7-9). Dwyer explains:

> **Another advantage gained with register renaming is the ability to overbook physical register assignment in a multi-threaded architecture. Many instruction set threads consist of only a fairly short number of instructions and only access a limited subset of the full architectural register set before they terminate, while some threads access the entire register set. Therefore, the average number of registers required by each thread is less than the full architectural set of registers. By sharing a pool of physical registers for renaming the architectural registers of several threads, a smaller physical register file can appear to provide full register sets to multiple threads. (Emphasis added, col. 2, lines 46-57)**

Dwyer discloses register mapping that enables each thread to access physical registers from the pool of registers that are different from the registers accessed by another thread. At no point however, does Dwyer discloses or suggests that any one physical register in the pool of

register is absolutely addressable by two or more threads. Dwyer, therefore, fails to disclose or suggest "each register in the plurality of windows of registers being relatively addressable by the corresponding thread and absolutely addressable by two or more of the threads executing on the multithreaded processor," as required by Appellant's independent claim 1.

Because neither Panwar nor Dwyer discloses or suggests, alone or in combination, at least the features of "each register in the plurality of windows of registers being relatively addressable by the corresponding thread and absolutely addressable by two or more of the threads executing on the multithreaded processor," Appellant's independent claim 1 is therefore patentable over these cited references.

No Reason to Combine Dwyer with Panwar

Even assuming, for argument sake, that Dwyer discloses the features the Examiner contended are disclosed in Dwyer, Appellant submits that in any event a person of ordinary skill in the art would have no reason to combine the teachings of the Dwyer reference with the teachings of the Panwar reference.

The Examiner stated in the February 20, 2007, Final Action that:

> It would have been obvious to one of ordinary skill in the art to use Dwyer in Panwar for including the register addressable by two or more threads executing on the multiprocessor as clamed because the use of Dwyer could provide Panwar the ability to accept data from more than one thread process, and thereby increasing the ability to adapt to multiple bandwidth of maltreated tasks, and it could be achieved by predefining the multiple threads of Dwyer into the configuration file of Panwar with modified control parameters (e.g. the register map with the corresponding threads) so that multiple threads of Dwyer could be recognized by the register of Panwar, and because Panwar also taught a parallel multithreaded processor (see consistent implementation of multithreaded processor 102 in col.6, lines 40-44),which was a suggestion of the need for providing an access to a register, or the like, by a multiplicity of threads, and in doing so, provided a motivation. (Final Action, page 5)

As explained above, Panwar discloses a processor that speculatively executes instructions that specify logical addresses, but, as also admitted by the Examiner, does not disclose a register addressable by two or more threads executing on a multiprocessor.

Panwar indicates that a motivation for defining register windows is to partition the aggregate of available registers into some pre-determined number of contiguous registers so that "[w]ith register windowing, a register window has a predetermined number of contiguous registers, and the window can be moved linearly within the register file. At any one time, the register window permits program access to a subset of the total number of registers in the register file" (col. 2, lines 44-48) Panwar also explains:

> **Referring to FIG. 3, a register file 300 having 128 registers is shown with a window 302 having 32 registers. Window 302 is movable within the register file 300 by a program or process executing on processor 102. For example, different processes running within processor 102 could allocate their own register window 302 to access 32 registers independent of the other processes executing within the processor. (Emphasis added, Panwar, col. 7, lines 40-47)**

Thus, a process using one register window on Panwar's apparatus runs independently of other processes and therefore has no need to access a register in some other register window.

Accordingly, even assuming that Dwyer discloses registers that are addressable by two or more threads, which Appellant contends Dwyer does not disclose, there would be no reason to combine Panwar, which calls in its implementation for registers that are arranged into windows of registers so that processes may easily access only the register windows they require, with a feature that enables two or more threads (or programs) to access a particular register. Indeed, at no point does Panwar suggest that a particular program accessing the various register window would require accessing a register not located in the window associated with that particular program.

Because no reason exists for combining Dwyer with Panwar, Appellant's submits that for this reason also independent claim 1 is patentable over the combination of Panwar and Dwyer.

For the foregoing reasons, Appellant submits that independent claim 1, and the claims depending from it, are patentable over the cited art.

Independent claims 12 and 21 recite "each register in the plurality of windows of registers being relatively addressable by the corresponding thread associated with the respective window of registers and absolutely addressable by two or more of the threads executing on the multi-threaded processor," or similar language. For reasons similar to those provided with

respect to independent claim 1, at least this feature is not disclosed by the cited art. Accordingly, independent claims 12 and 21, and the claims that depend from them, are patentable over the cited art.

**(3) Claim 10 is patentable over the prior art**

Appellant's claim 10, which depends from independent claim 1, recites the features of "wherein the exact address of the register is directly specified in a source field or destination field of the instruction." The Examiner rejected claim 10 as anticipated by Sollars and also as being rendered obvious over the combination of Panwar and Dwyer.

Particularly, with respect to the Examiner's rejection of claim 10 as anticipated by Sollars, the Examiner stated:

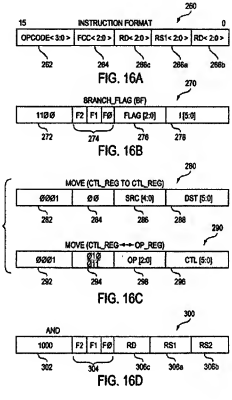
> 43. As to clam 10, since Sollars also taught his registers sets are operand registers, therefore, the source and destination must be included, such as source opened or destination operand.

Sollars describes:

> **Besides modifications resulting from the normal course of instruction execution, the control registers of primary control register file 20a are directly accessible and modifiable using instructions of the standard instruction set of processor 10. FIGS. 16a-16d illustrate the presently preferred embodiment of the instruction format for instructions of the standard instruction set, and three selected ones of these instructions. As shown in FIG. 16a, each instruction includes integrated functional unit configuration control information (FCC) 264 for dynamically configuring applicable functional units to perform different variations of functions and/or different functions. While FCC 264 complements opcode 262, FCC 264 of each instruction can be provided to the applicable functional unit without requiring IFU 12 to perform any decoding. As a result, instruction decoding by IFU 12 is further speeded up.**
>
> **As shown in FIGS. 16b-16d, the standard instruction set includes, in particular, a branch-on-flag instruction 122, a "MOV" instruction 280/290, and an AND instruction 300. Branch-on-flag instruction 122 is used for effectuating instruction branching based on the values of the hardware/software flags in HWFLAG and SWFLAG control registers 112e and 112f of a thread. MOV instruction 280/290 is used for moving "data" between control registers, as well as between control and operand registers. The two like kinds of moves are implemented with the same common "MOV" instruction, i.e., same opcode 262. The differentiation of the two like kinds of moves is denoted by integral FCC information 284/294.**

**Similarly, AND instruction 300 is used for performing either an AND or a NOR operation, depending on integral FCC information 304. (Sollars, col. 14, lines 37-67)**

FIGS 16a-16d discussed in the above passage show:




While Sollars uses fields in instructions to specify registers, at no point does Sollars describe that exact addresses of registers are specified in those instructions. Appellant further notes that operand registers (which the Examiner cites in support of the argument that Sollars discloses instructions that specify an exact address) are merely registers that hold data to be used in operations performed by instructions (e.g., "MOV instruction 280/290 is used for moving "data" between control registers, as well as between control and operand registers.", col. 14, lines 59-61).

Appellant therefore contends that Sollars does not disclose or suggest the claim 10 feature of "wherein the exact address of the register is directly specified in a source field or destination field of the instruction".

As for the Examiner's rejection of claim 10 as rendered obvious by Panwar And Dwyer, the Examiner stated:

> 22. As to claim 10, Panwar also included source field and destination field (see destination and source in col. 2, lines 66-67, col.6, lines 1-5). (Final Action, page 6)

Appellant contends that the Examiner characterization of Panwar is incorrect and that Appellant's claim 10 is patentable over the cited art.

As explained above, Panwar discloses programs executing on a processor 102 that access registers through a typical naming convention, such as r0, r1, r2, …, r30 (col. 7, lines 51-54). As further explained, Panwar does not describe instructions that specify a register's absolute address. With respect to the passages alluded to by the Examiner, these passages are reproduced below:

> **The five-bit register addresses encoded in an instruction word specify the instruction's source registers and the destination register. These register specifiers are logical addresses that index registers within the current register window. (Emphasis added, col. 2, line 66 – col. 3, line 3)**

And:

> **In the absence of conditional branch instruction, IFU 202 addresses the instruction cache sequentially. The branch prediction logic in IFU 202 handles branch instructions, including unconditional branches. An outcome tree of each branch instruction is formed using any of a variety of available branch prediction algorithms and mechanisms. More than one branch can be predicted simultaneously by supplying sufficient branch prediction resources. After the branches are predicted, the address of the predicted branch is applied to the instruction cache rather than the next sequential address. If a branch is mispredicted, the instructions processed from the mispredicted branch are flushed from the processor, and the processor state is restored to the state prior to the mispredicted branch. For instructions which affect the speculative calculation of the physical address of a register, restoration of the processor's window management registers is discussed below with reference to FIGS. 6-7. (col. 5, line 56, to col. 6, line 6)**

Thus, Panwar explicitly states that the five-bit register addresses encoded in an instruction word specify logical addresses (i.e., relative addresses.) Panwar uses logical address to calculate physical addresses, but no where does Panwar describes that registers' physical addresses are encoded in instructions.

Dwyer, as explained above, uses register maps to map between threads' architectural register references (i.e., relative addresses) and physical locations of registers. Dwyer too, therefore, does not describe that registers' physical addresses are specified in threads' instructions.

Accordingly, because none of the references discloses or suggests, alone or in combination, at least the features of "wherein the exact address of the register is directly specified in a source field or destination field of the instruction," Appellant contends that claim 10 is patentable over the cited art.

**Conclusion**

For the foregoing reasons, Appellant submits that Claims 1-8 and 10-21 are allowable. Therefore, the Examiner erred in rejecting Appellant's claims and should be reversed.

Respectfully submitted,

Date: Sept. 28, 2007

Ido Rabinovitch
Attorney for Intel Corporation
Reg. No. L0080

PTO Customer No. 20985
Fish & Richardson P.C.
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

## Appendix of Claims

1. A method of maintaining execution threads in a parallel multithreaded processor comprises:

accessing, by a thread executing in the multithreaded processor, a register in a register set organized into a plurality of windows of registers, each of the plurality of windows of registers associated with a corresponding thread, each register in the plurality of windows of registers being relatively addressable by the corresponding thread and absolutely addressable by two or more of the threads executing on the multithreaded processor with absolutely addressable comprises providing an exact address of the register with the exact address specified in an instruction.

2. The method of claim 1 wherein multiple threads can use the same control store and relative register locations but access different window banks of registers.

3. The method of claim 1 wherein the relative register addressing divides the register banks into windows across the address width of the general purpose register set.

4. The method of claim 1 wherein relative addressing allows access any of the window registers relative to the starting point of a window of registers.

5. The method of claim 1 further comprising:

organizing the register set into windows according to the number of threads that execute in the processor.

6. The method of claim 1 wherein relative addressing allow the multiple threads to use the same control store and locations while allowing access to different windows of register and perform different functions.

7. The method of claim 1 wherein the window registers are implemented using dual ported random access memories.

8. The method of claim 1 wherein relative addressing allows access to any of the windows of registers relative to the starting point of the window of registers.

9. (Cancelled)

10. The method of claim 1 wherein the exact address of the register is directly specified in a source field or destination field of the instruction.

11. The method of claim 1 wherein relative addresses are specified in instructions as an address offset within a context execution space as defined by a source field or destination field operand.

12. A hardware based multi-threaded processor comprises:

a processor unit comprising:

control logic circuit including context event switching logic, the context switching logic arbitrating access to the microengine for a plurality of executable threads;

an arithmetic logic unit to process data for executing threads; and

a register set that is organized into a plurality of windows of registers, each of the plurality of windows of registers associated with a corresponding one of the plurality of threads, each register in the plurality of windows of registers being relatively addressable by the corresponding thread associated with the respective window of registers and absolutely addressable by two or more of the threads executing on the multi-threaded processor, with any one of the registers of the register set being absolutely addressable by providing an exact address of the register with the exact address specified in an instruction.

13. The processor of claim 12 wherein the control logic circuit further comprises:

an instruction decoder; and

program counter units to track executing threads.

14. The processor of claim 13 wherein the program counters units are maintained in hardware.

15. The processor of claim 12 wherein the register set is organized into windows across an address width of the general purpose register set with each window relatively accessible by the corresponding thread.

16. The processor of claim 15 wherein the relative addressing allows access to any of the registers relative to the starting point of a window of registers.

17. The processor of claim 15 wherein the number of windows of the register set is according to the number of threads that execute in the processor.

18. The processor of claim 12 wherein relative addressing allow the multiple threads to use the same control store and locations while allowing access to different windows of register and perform different functions.

19. The processor of claim 12 wherein the windows of registers are provided using dual ported random access memories.

20. The processor of claim 12 wherein the processing unit is a microprogrammed processor unit.

21. A computer program product residing on a computer readable storage medium for managing execution of multiple threads in a multithreaded processor comprising instructions causing a processor to:

access, by an executing thread in the multithreaded processor, a register in a register set organized into a plurality of windows of registers, each of the plurality of windows of registers being associated with a corresponding thread, each register in the plurality of windows of registers being relatively addressable by the corresponding thread and absolutely addressable by

two or more threads executing on the multithreaded processor with absolutely addressable comprises instructions that when executed cause the processor to provide an exact address of the register with the exact address specified in an instruction.

22. (Cancelled)

**EVIDENCE**

**APPENDIX**

None

## RELATED PROCEEDINGS APPENDIX

None